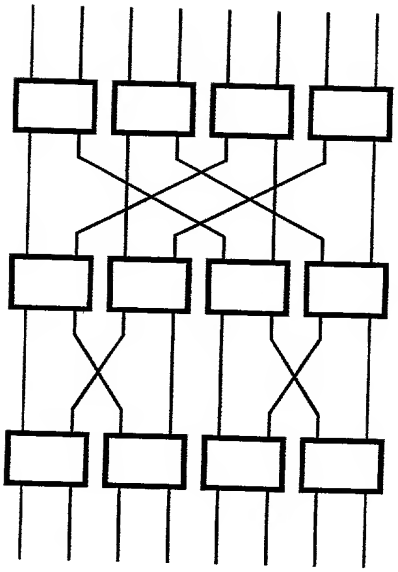


Fig. 1A (prior art)

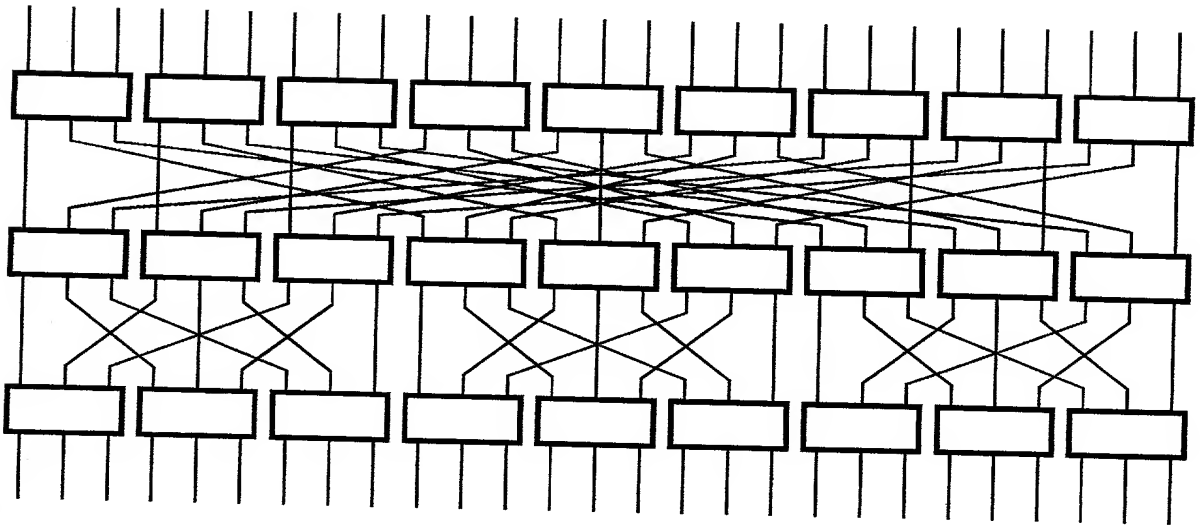


Fig. 1B (prior art)

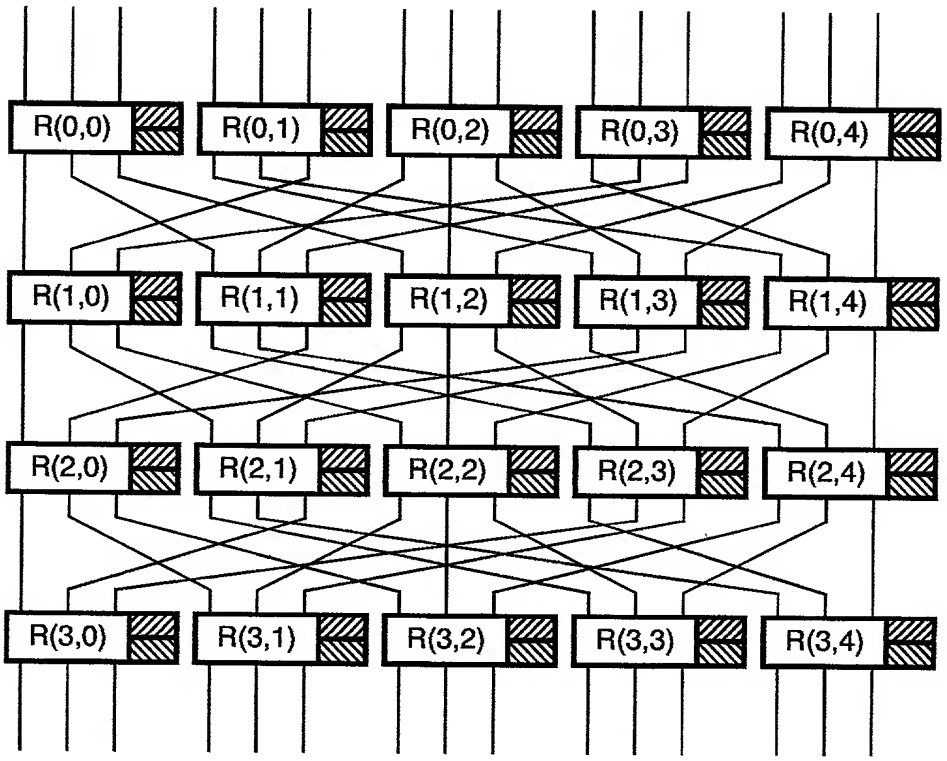



Fig. 2

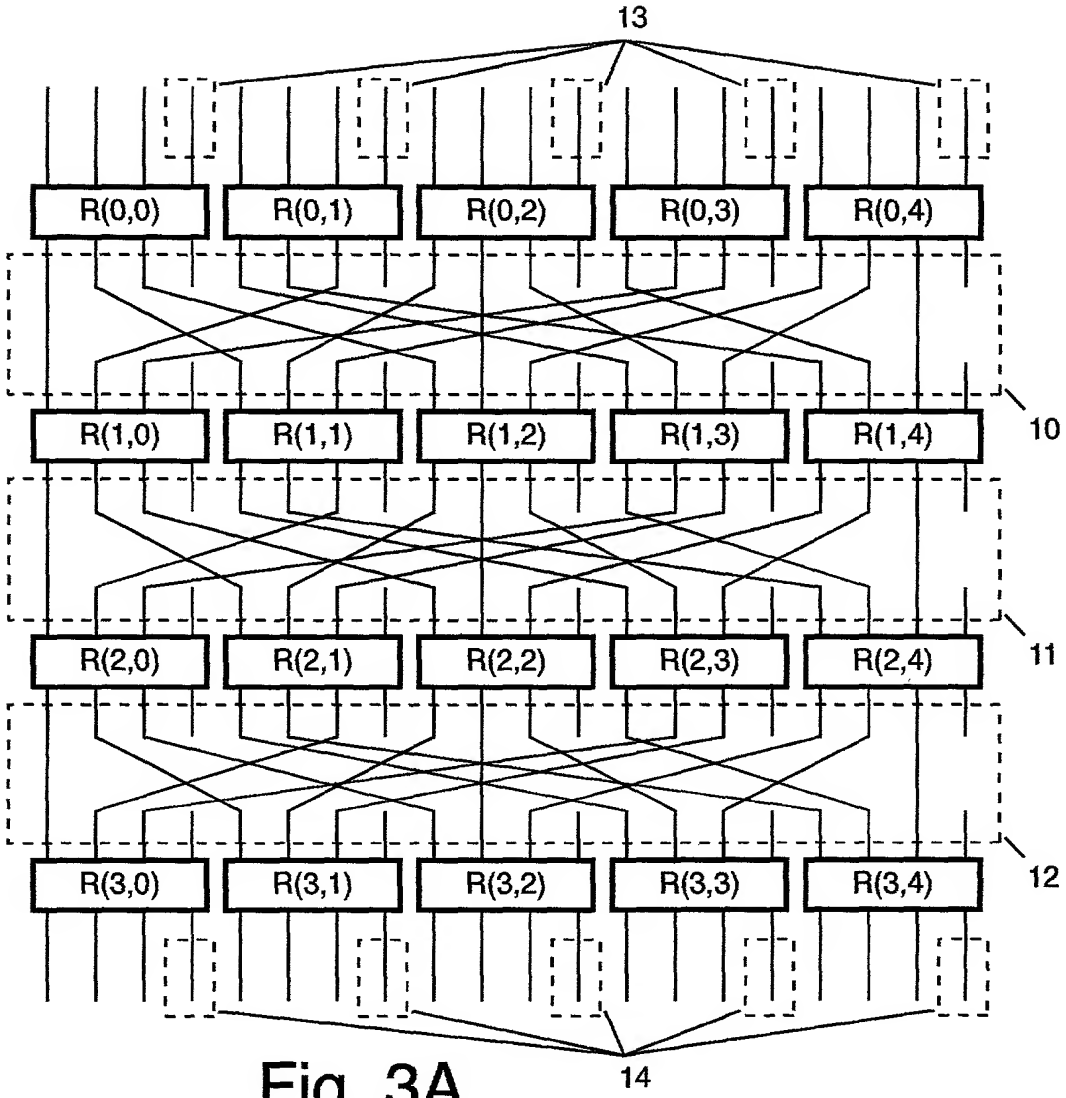



Fig. 3A

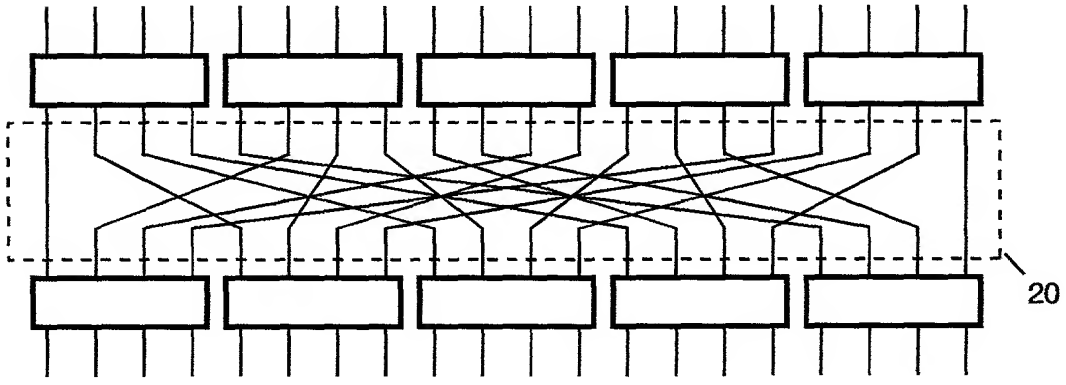



Fig. 3B

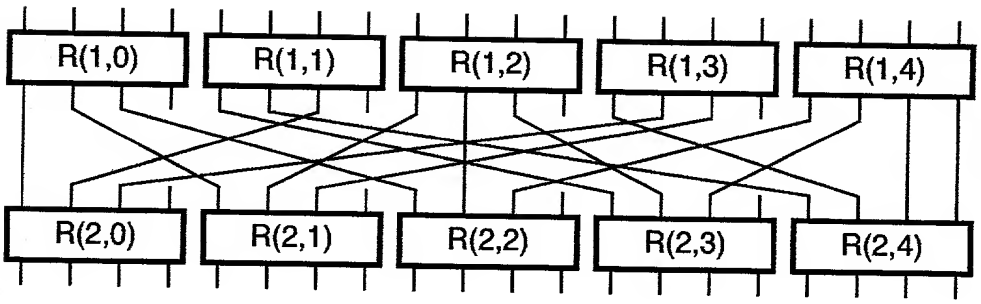



Fig. 4A

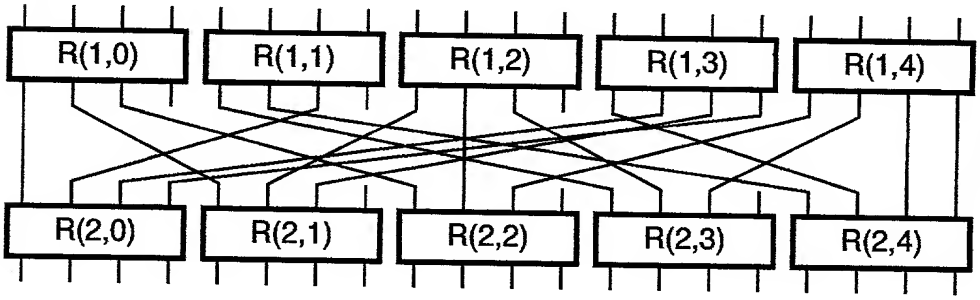



Fig. 4B

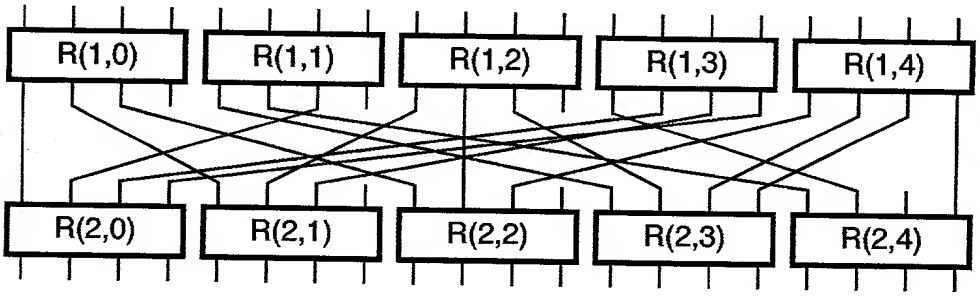



Fig. 4C

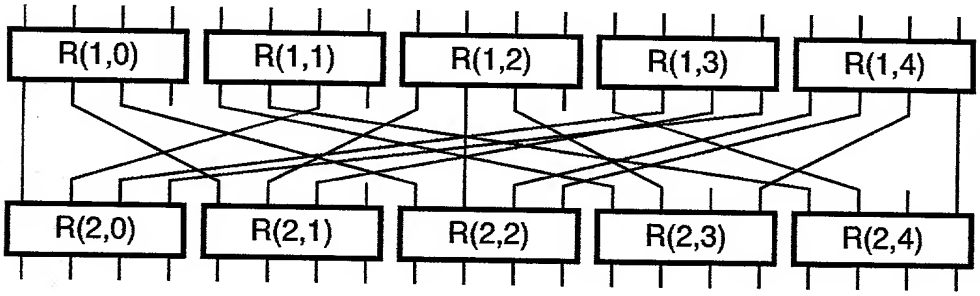



Fig. 4D

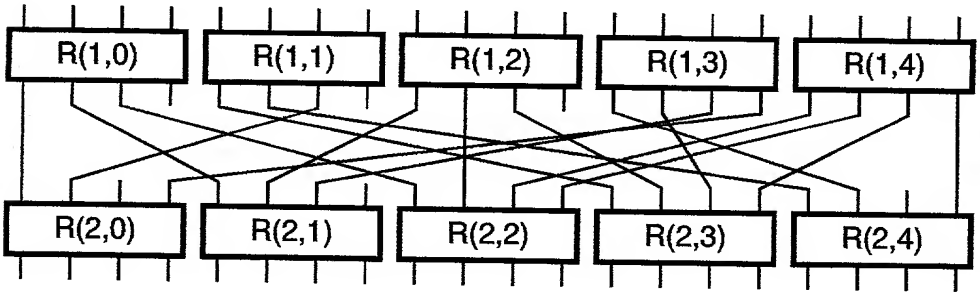



Fig. 4E

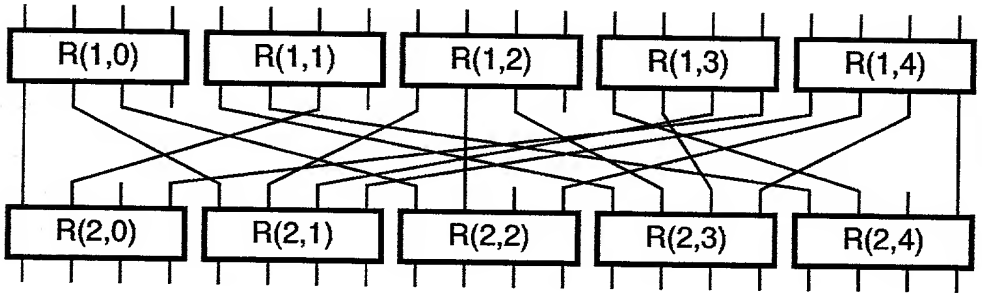



Fig. 4F

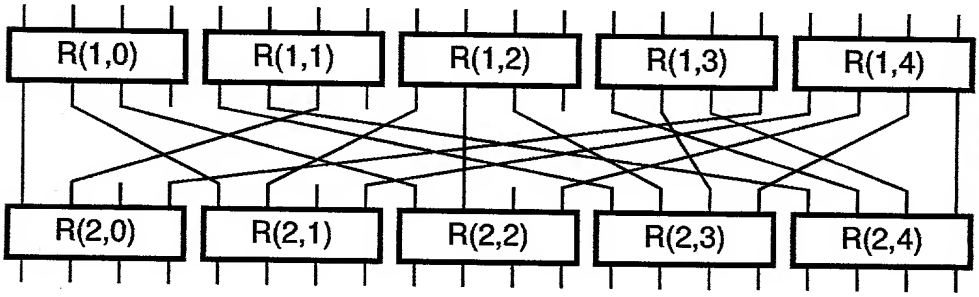



Fig. 4G

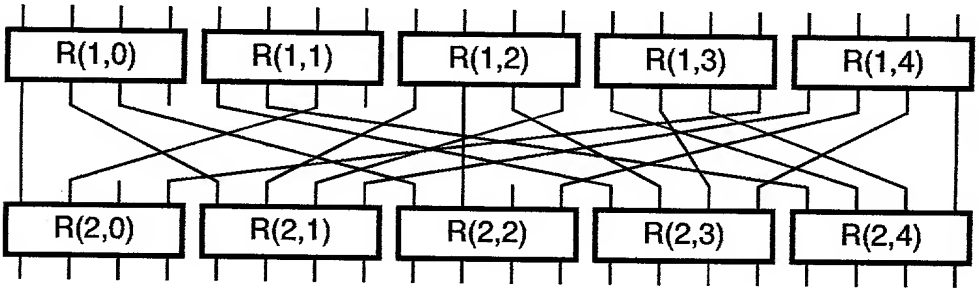



Fig. 4H

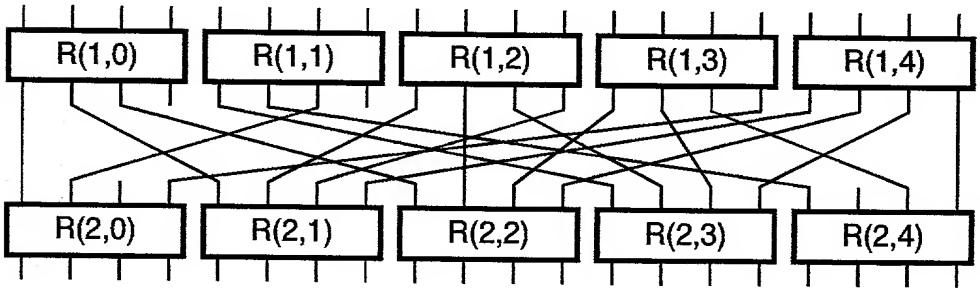



Fig. 4I

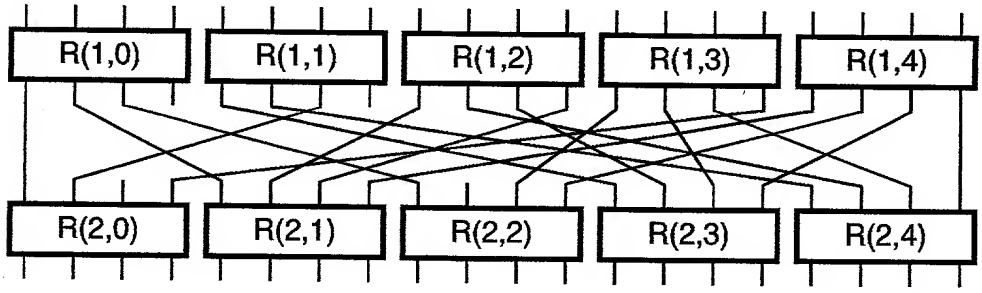



Fig. 4J

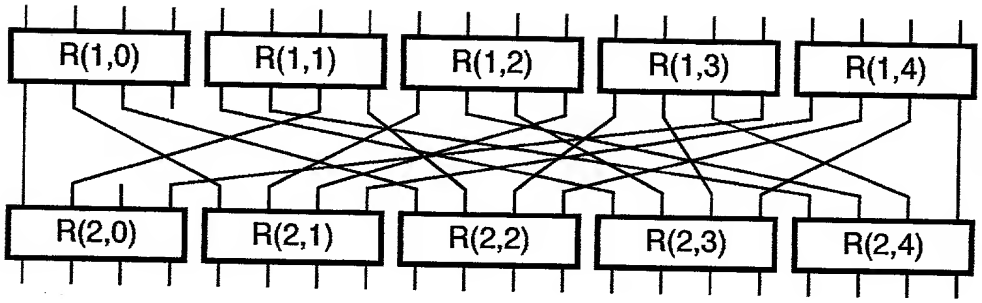



Fig. 4K

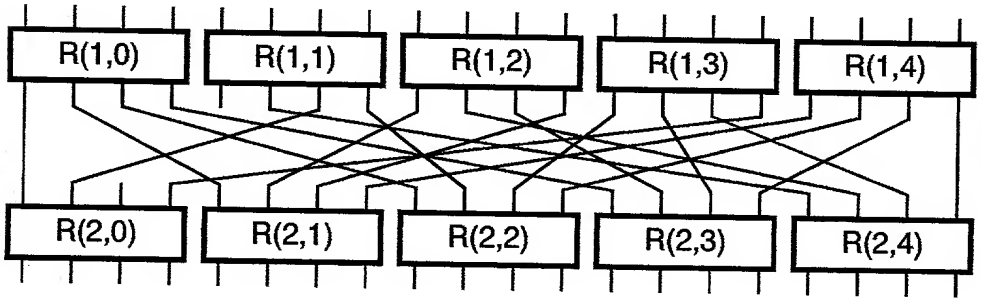



Fig. 4L

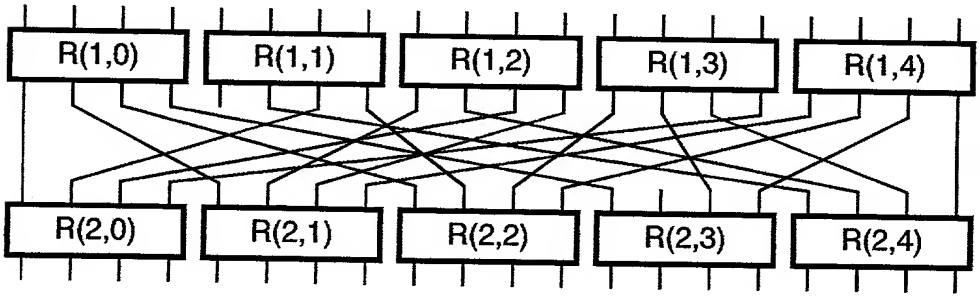



Fig. 4M

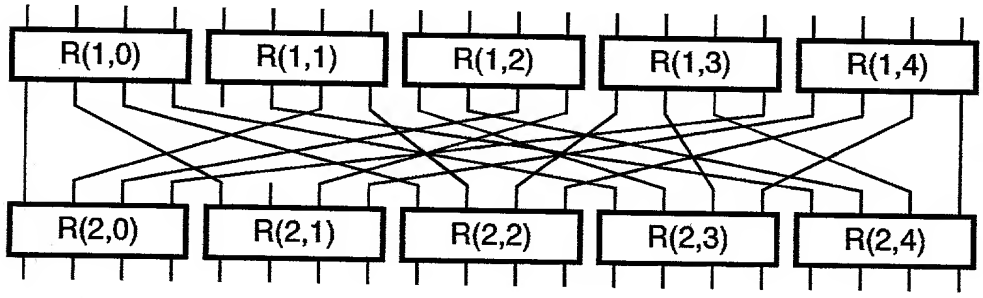



Fig. 4N

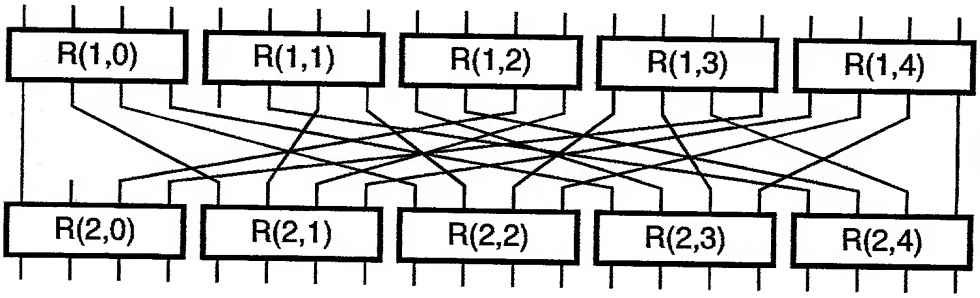



Fig. 4O

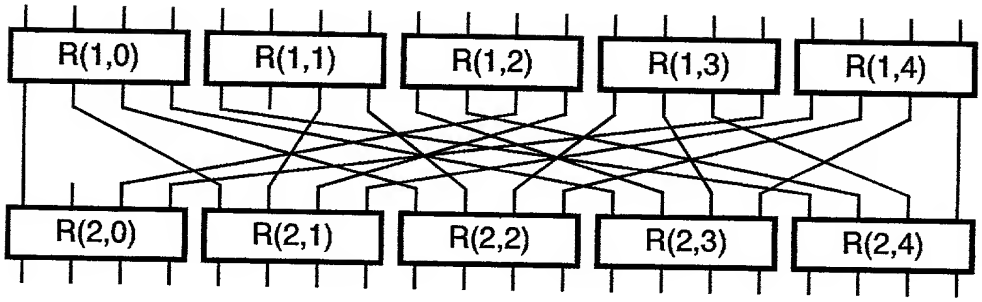

R(1,0)
R(1,1)
R(1,2)
R(1,3)
R(1,4)
R(2,0)
R(2,1)
R(2,2)
R(2,3)
R(2,4)

Fig. 4P

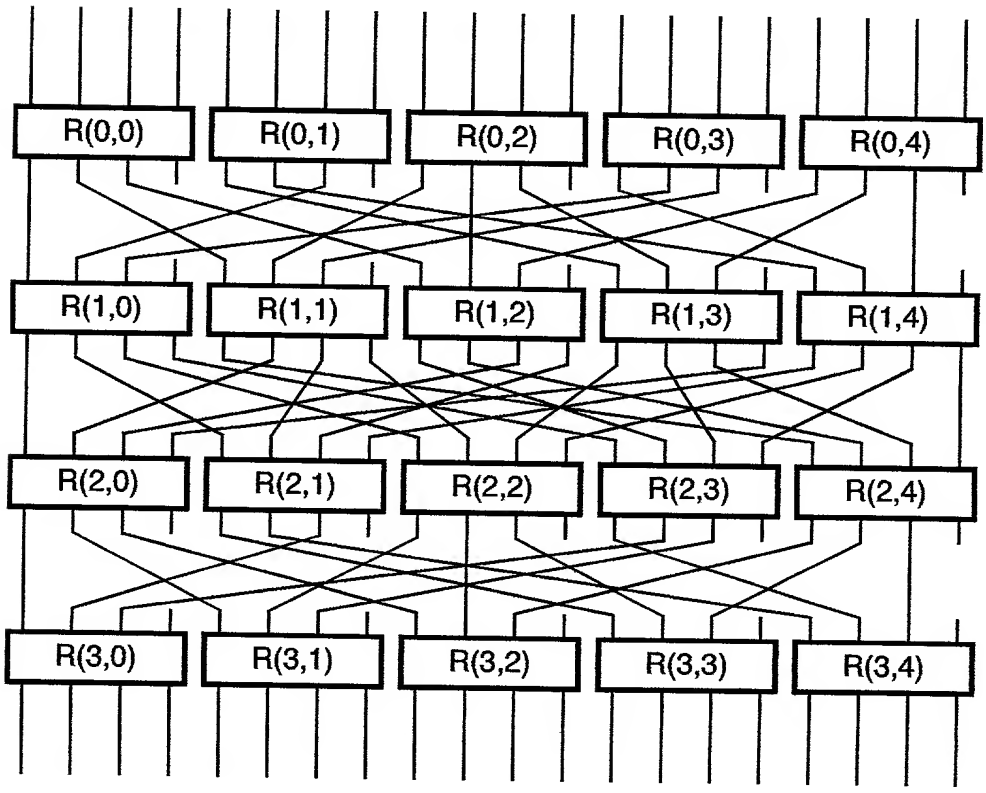

R(0,0)
R(0,1)
R(0,2)
R(0,3)
R(0,4)
R(1,0)
R(1,1)
R(1,2)
R(1,3)
R(1,4)
R(2,0)
R(2,1)
R(2,2)
R(2,3)
R(2,4)
R(3,0)
R(3,1)
R(3,2)
R(3,3)
R(3,4)

Fig. 4Q

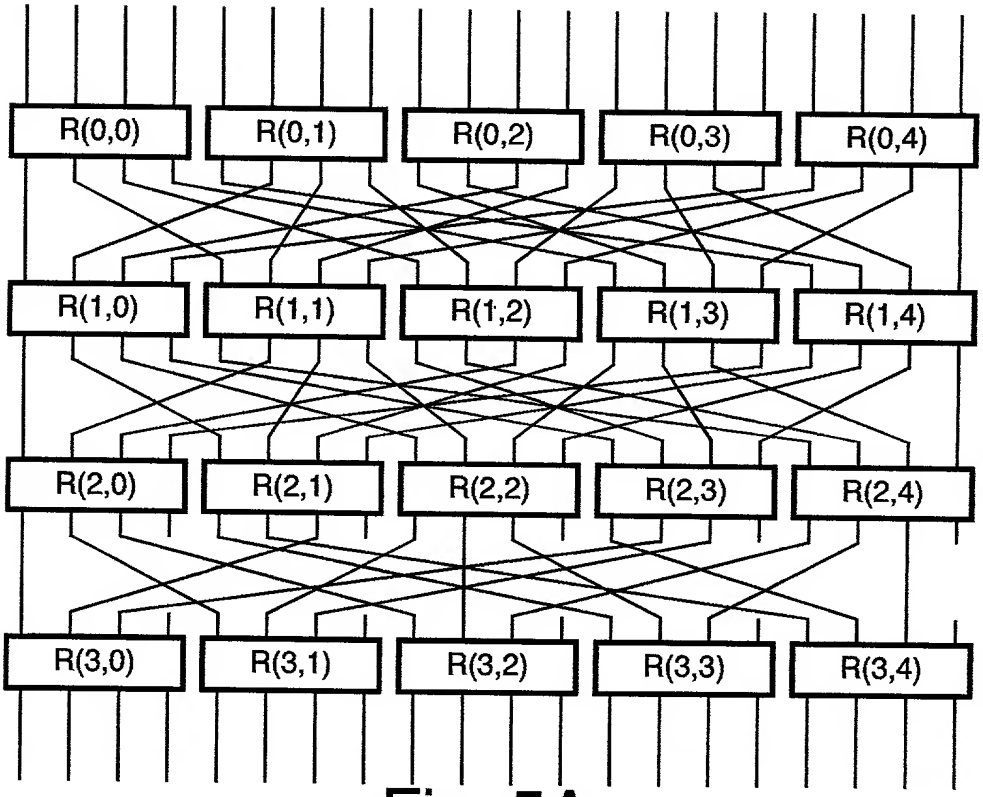



Fig. 5A

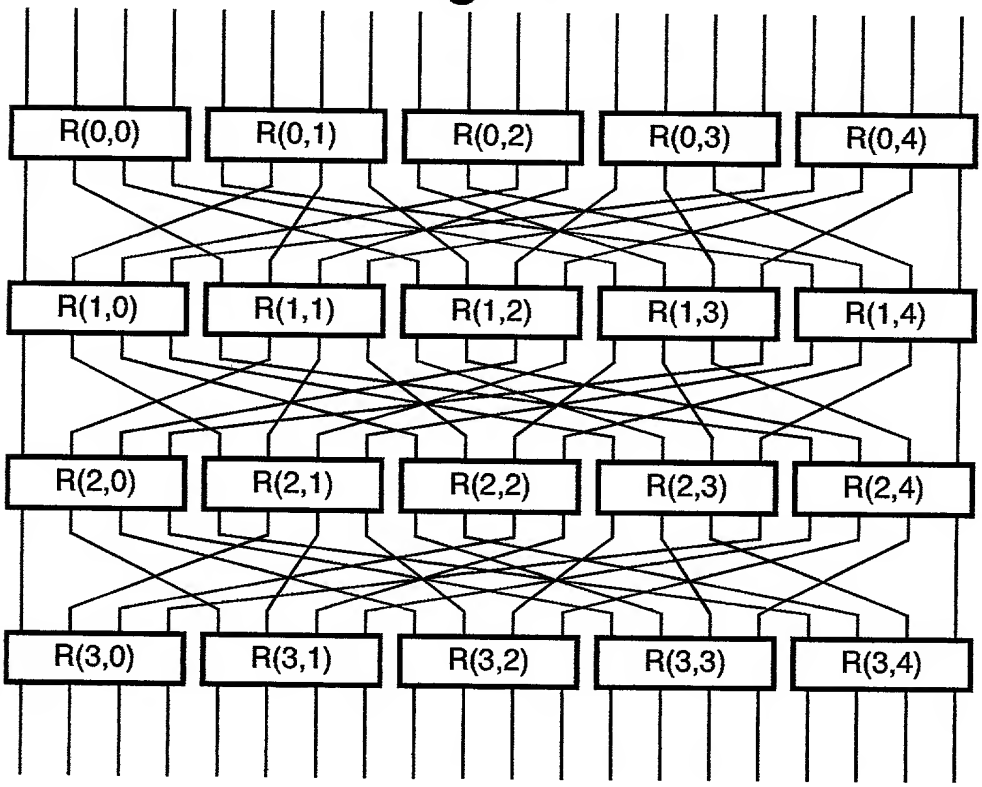



Fig. 5B

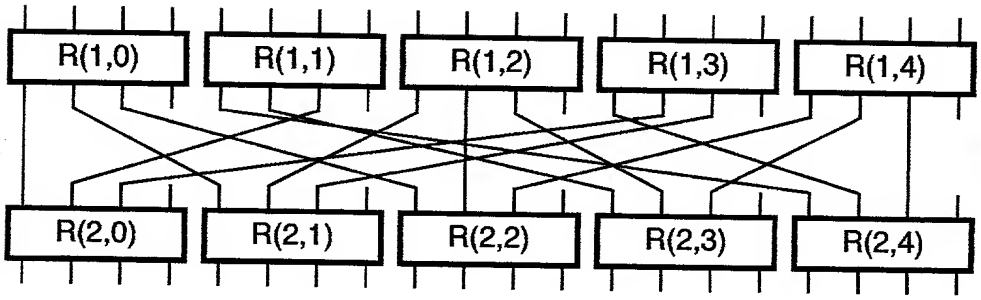



Fig. 6A

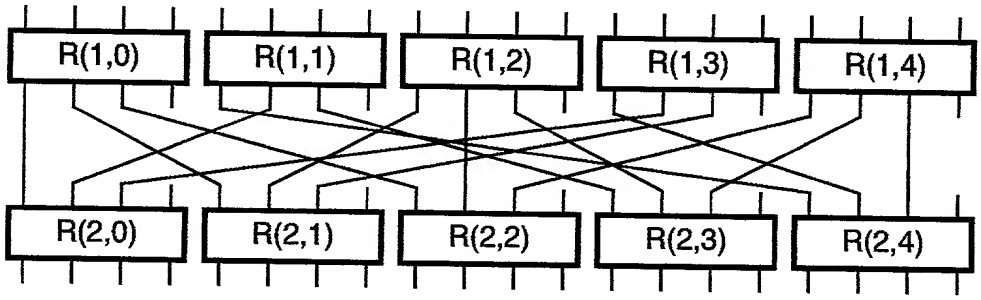



Fig. 6B

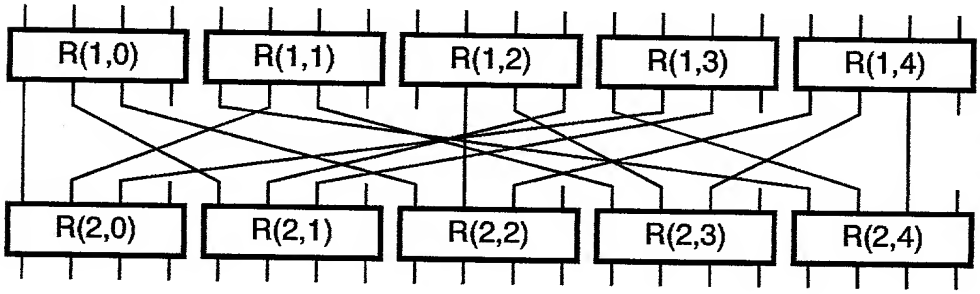



Fig. 6C

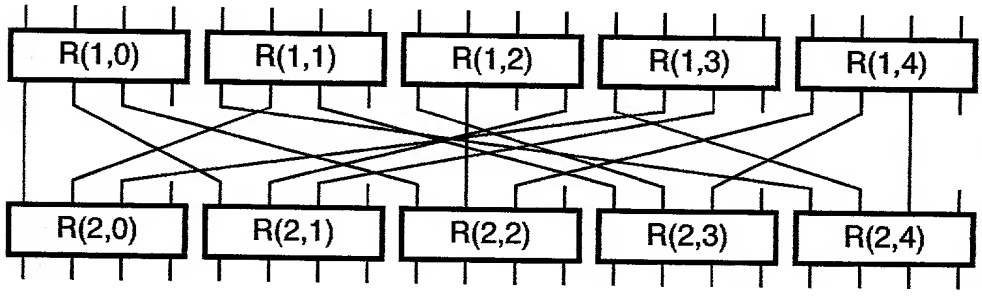



Fig. 6D

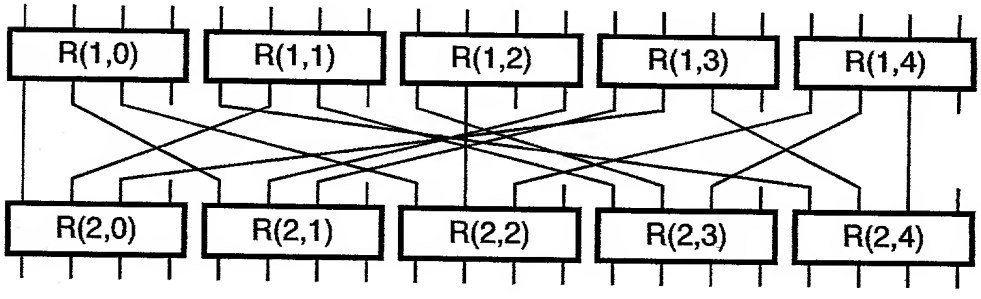



Fig. 6E

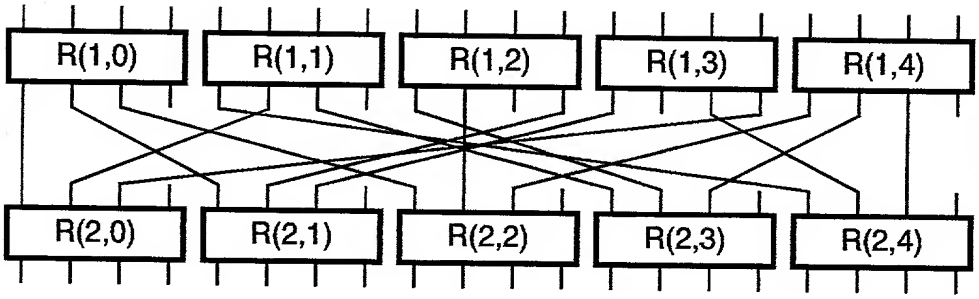



Fig. 6F

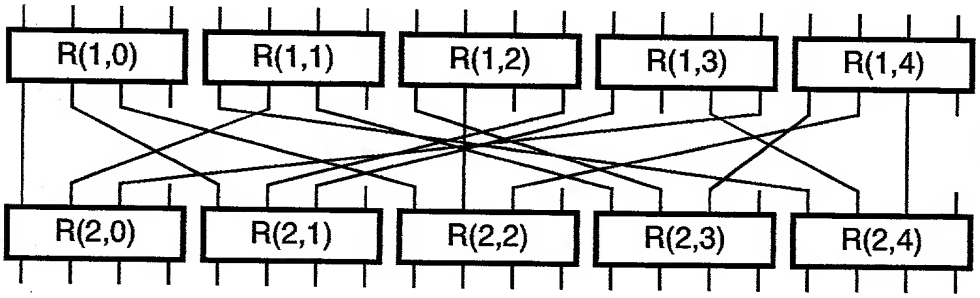



Fig. 6G

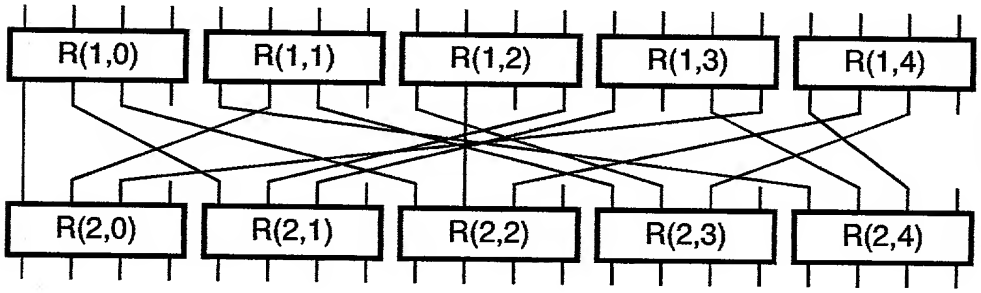



Fig. 6H

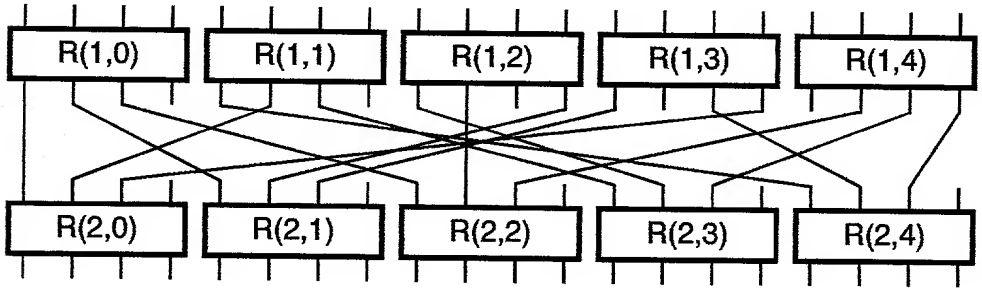



Fig. 6I

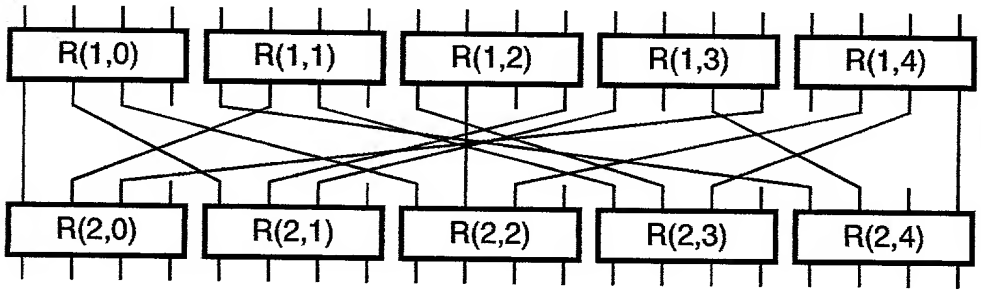



Fig. 6J

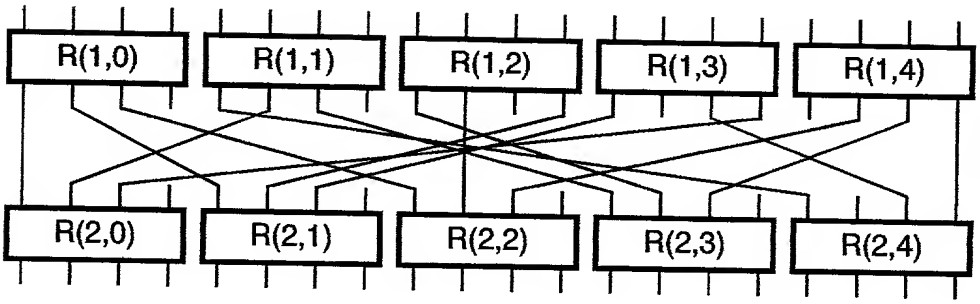



Fig. 6K

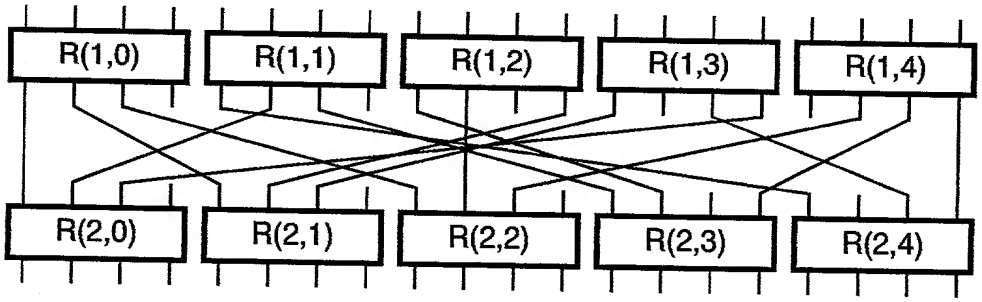



Fig. 6L

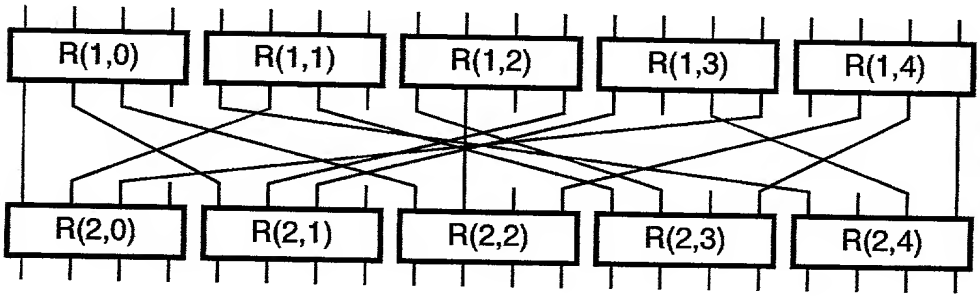



Fig. 6M

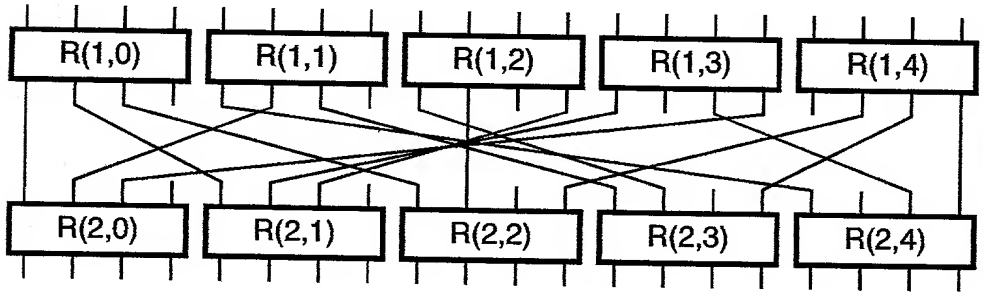



Fig. 6N

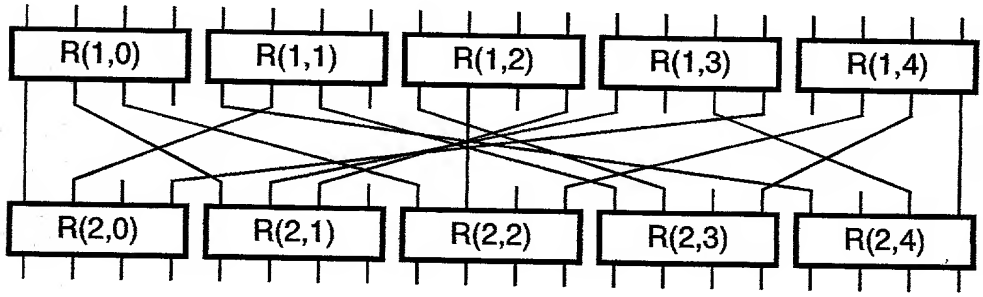



Fig. 6O

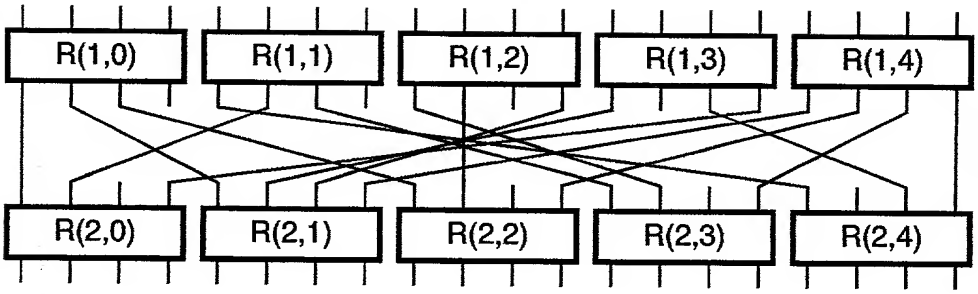



Fig. 6P

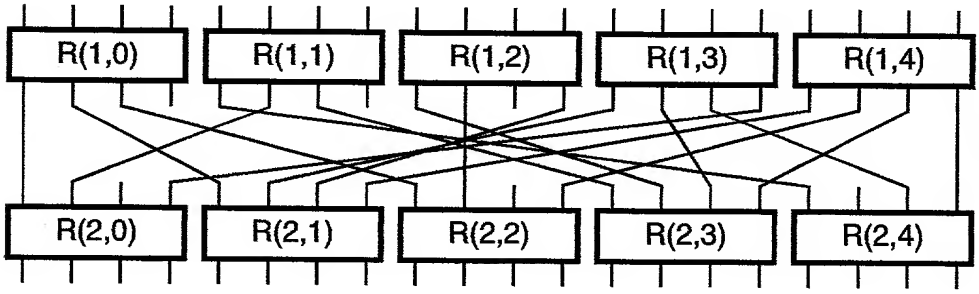



Fig. 6Q

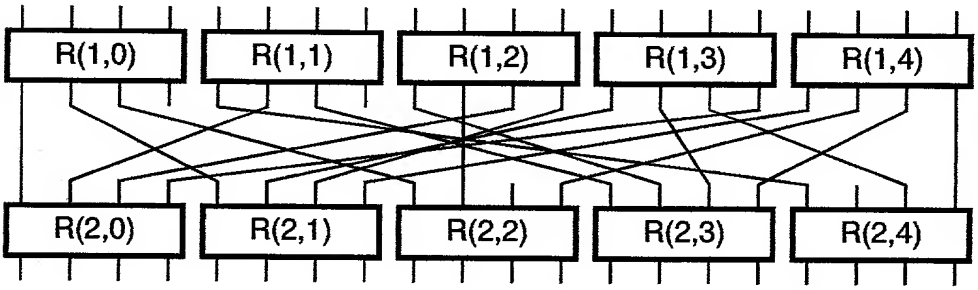



Fig. 6R

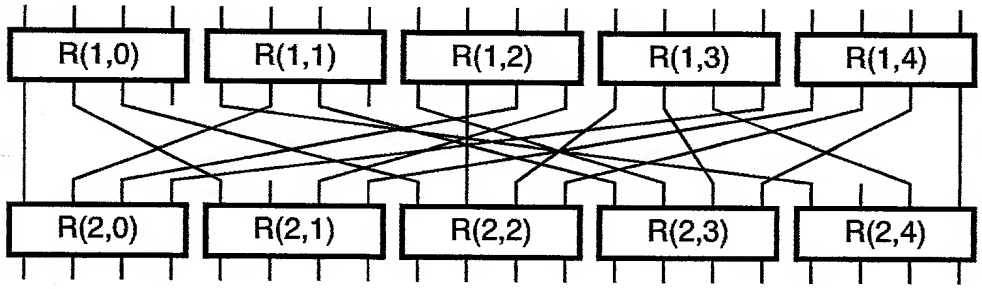



Fig. 6S

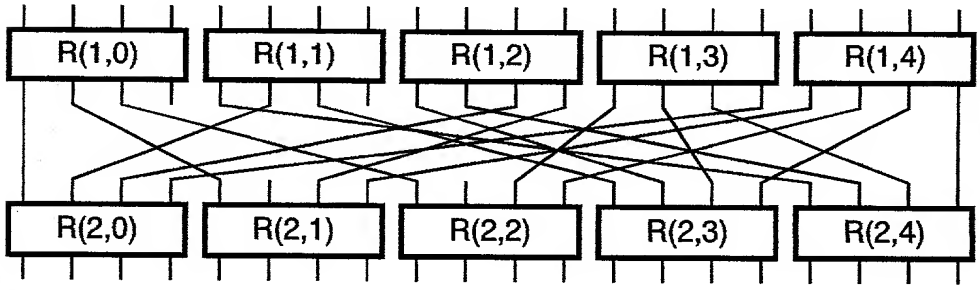



Fig. 6T

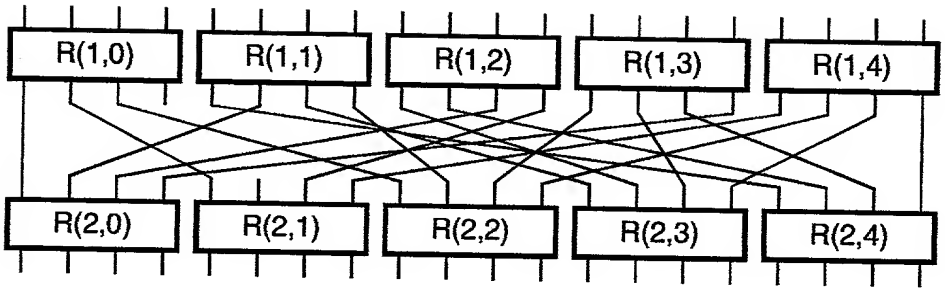



Fig. 6U

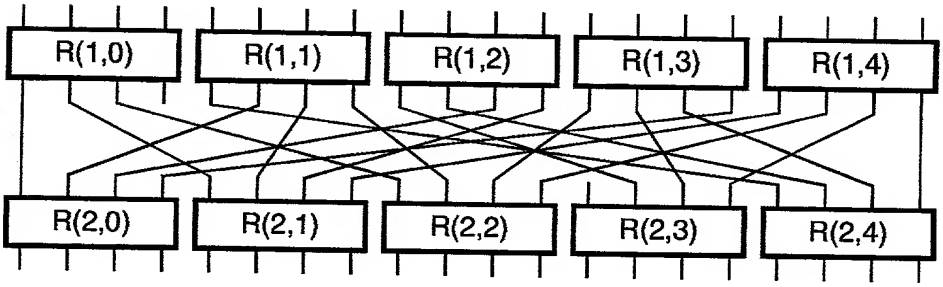



Fig. 6V

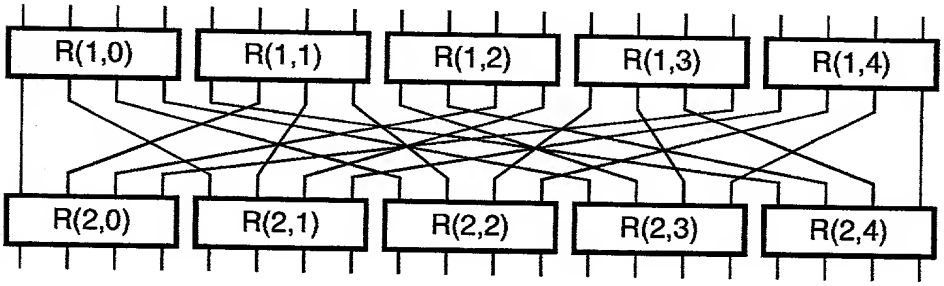



Fig. 6W

```
//
//Procedure Upgrade
//      Performs a fanout upgrade of a network.
//      "want to relabel" may be "true" or "false" each time the statement is reached
//
Procedure Upgrade
      do {
            select any router requiring addition of ports
            add_ports(router);
      } while (there are routers requiring new ports);
      do {
      start:
            if(want to relabel) {
                  if(any router, current_router, can be relabeled) {
                        relabel_ports(current_router);
                        goto start;
                  }
            }
            select any port not connected to corresponding_port(port);
            target_port=corresponding_port(port);
            if(target_port is already connected) {
                  disconnected_port=port currently connected to target_port;
                  Disconnect(target_port,disconnected_port);
            }
            Connect(port,target_port);
      } while(there are misconnected ports);
connect_external_ports();
activate_external_ports();
```

Fig. 7A

```
//
//Procedure Upgrade
//      Performs a fanout upgrade of an RCCBG network with an upgraded fanout of fanout,
//      num_routers_per_row per row, and num_rows total rows. Also,
//      RELABEL_AVAILABLE flag if swapping of ports in a single router can be performed without
//      breaking connections.
//
Procedure Upgrade
do {
      select any router requiring addition of ports
      add_ports(router);
} while (there are routers requiring new ports);

for(rindex=0;rindex<num_rows-1;rindex++) {
      current_row=row_select(rindex);
      if(RELABEL_AVAILABLE) {
            relabel_ports(current_row);
      }
      disconnected_port=None; // Holds the port previously disconnected by the last rewire step
      while((port=select_port(disconnected_port,current_row))!=None) {
            target_port=corresponding_port(port);
            if(target_port is already connected) {
                  disconnected_port=port currently connected to target_port;
                  Disconnect(target_port,disconnected_port);
            }
            Connect(port,target_port);
      }
connect_external_ports();
activate_external_ports();
```

Fig. 7B

```
//
//Simplification functions.
//
Function correct_port(port1,port2)
{
    if(port1 can be propertly connect to port2)return(TRUE);
    else return(FALSE);
}
Function corresponding_port(port)
{
    if(port is a bottom port) {
        return top port of router in next row that should be properly connected to port port;
    } else {
        return bottom port of router in the previous row that should be properly connected to port port;
    }
}
Function Disconnect(port1,port2)
{
    Divert traffic away from port1 ;
    Divert traffic away from port2 ;
    Disconnect connection between port1 and port2 ;
}
Function Connect(port1,port2)
{
    Connect port1 and port2 ;
    Allow traffic to flow through port1 ;
    Allow traffic to flow through port2 ;
}
```

Fig. 7C

```
Function row_select(row_index) {
    if(num_rows is even) {
        start_row=num_rows/2-1;
    } else {
        start_row=(num_rows-1)/2;
    }
    if(row_index is even) {
        return(start_row+row_index/2);
    } else {
        return(start_row-(row_index+1)/2);
    }
}
```

Fig. 8A

```
Function row_select(row_index) {
    return(row_index);
}
```

Fig. 8B

```
Function select_port(dport,current_row) // optimal dport is not used
{
    port_pool={port: bottom ports of routers in row, current_row and top port of routers in row,
                current_row+1 not connected to corresponding_port(port)};
    // For simplicity order right to left
    // First criterion
    for port in port_pool {
        if(disconnected(port) && disconnected(corresponding_port(port))return(port);
    }
    // Second criterion:This basically says we prefer to target connections that break
    // connections only on fully populated routers
    for port in port_pool {
        if(disconnected(port) &&
            num_disconnections(router_of(port_connected_to(corresponding_port(port)==0))) {
            return(port);
    }
    // Third criterion:Any port that is not connected
    for port in port_pool {
        if(disconnected(port)) return(port);
    }
    // Catch all for any ports left over:Not likely to be needed
    for port in port_pool {
        return(port);
    }
    return(None);
}
```

Fig. 9A

```
Function select_port(dport,row) //f ill the hole
{
    if(dport !=None)return(dport);
    else {
        for all bottom ports, port, of routers in row current_row scanning from right to left {
            if(port is not connected to corresponding_port(port)) return(port);
        }
        return None; // No more ports to rewire
    }
}
```

Fig. 9B

```
Function select_port(dport,current_row) // round robin
{
    // This requires a FIFO of ports
    if(port_fifo empty) {
        port_fifo={port: bottom ports of routers in row, current_row and top port of routers in row,
                current_row+1 which are disconnected};
    if(port_pool empty) {
        port_pool={port: bottom ports of routers in row, current_row and top port of routers in row,
                current_row+1 not connected to corresponding_port(port)};
        for port in port_pool {
            return(port);
        }
        // Catch all for any ports left over:Not likely to be needed
        port=any port not connected to proper port
        if(port exists) {
            return(port);
        }else {
            return(None);
        }
    }
    port=top of port_fifo ;
    remove top of port_fifo ;
    return(port);
}
```

Fig. 9C

```
Function relabel_ports(current_row)
{
    for(i=0;i<routers_per_row;i++) {
        for(bport1=0;bport1<fanout;bport1++) {
            for(bport2=0;bport2<fanout;bport2++) {
                //Test to see if the candidate port is connected to a router which one of the
                //other ports on the same router should be connected to.It doesn 't matter
                //at this point if it is the correct top port.That will be corrected in next loop.
            if(bottom port bport1 of R(current_row,i)is connected to any top port of
                router_of(corresponding_port(bottom port bport2 of R(current_row, i)) {
                if(bport1!=bport2) {
                    exchange_ports(bport1 of R(current_row, i),bport2 of R(current_row, i));
                }
            }
        }
    }
    for(tport1=0;tport1<fanout;tport1++) {
        for(tport2=0;tport2<fanout;tport2++) {
            //Test to see if the candidate port is connected to a port which one of the
            //other ports on the same router should be connected to.
            if(top port tport1 of R(current_row+1, i)is connected to
                corresponding_port(top port tport2 of R(current_row+1, i)) {
                if(tport1!=tport2) {
                    exchange_ports(tport1 of R(current_row+1, i),tport2 of R(current_row+1, i));
                }
            }
        }
    }
}
//
//Auxiliary Procedures
//
Function router_of(port)
{
    return(the router which port belongs to);
}
//
//Here logical relabelling is assumed possible
//Other exchange schemes can be substituted
//
Function exchange_ports(port1,port2)
{
    permanently divert traffic originally intended for port1 to port2 ;
    permanently divert traffic originally intended for port2 to port1 ;
}
```

Fig. 10

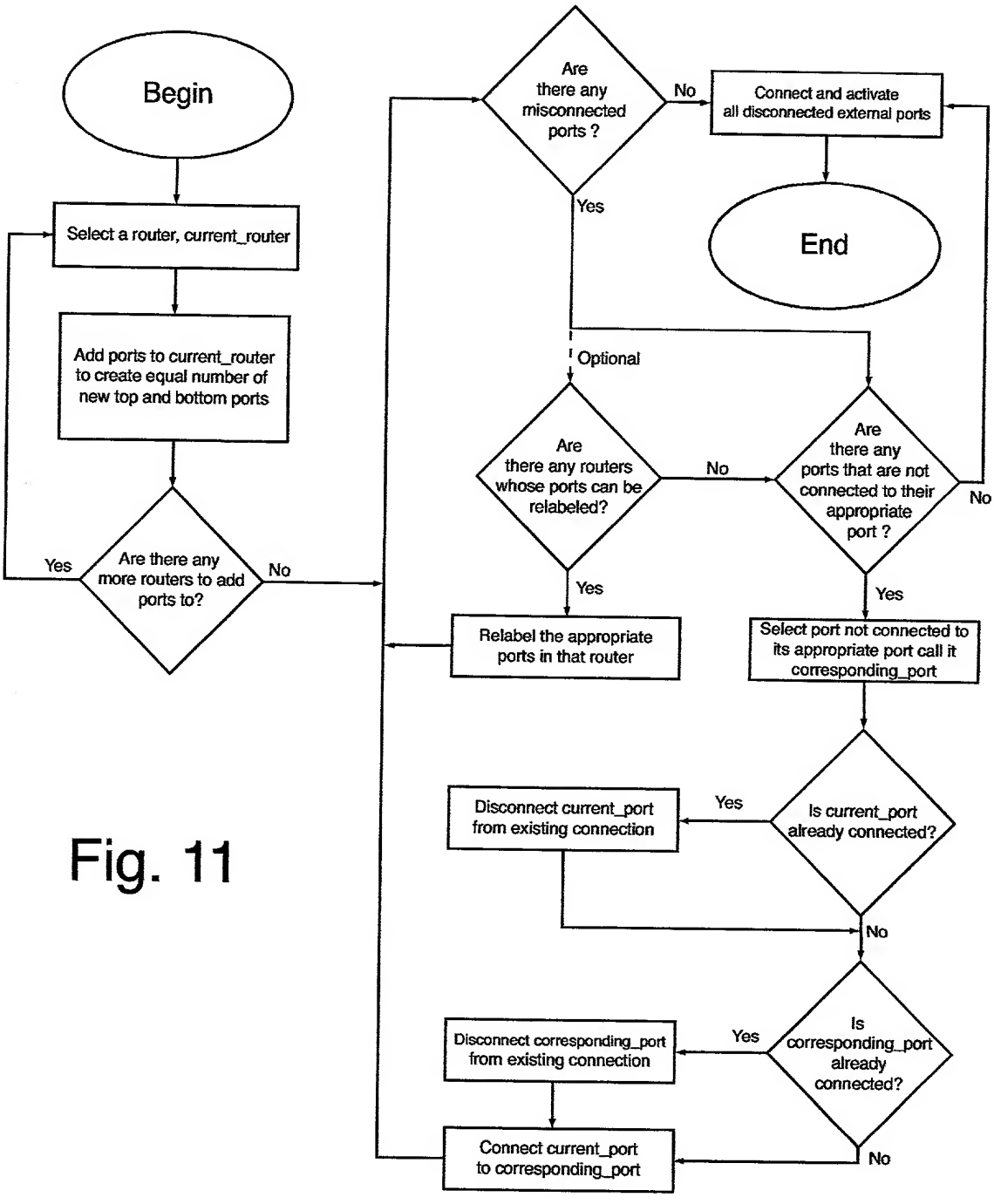

Begin
Select a router, current_router
Add ports to current_router to create equal number of new top and bottom ports
Are there any more routers to add ports to?
Yes
No
Are there any misconnected ports ?
No
Connect and activate all disconnected external ports
End
Yes
Optional
Are there any routers whose ports can be relabeled?
No
Are there any ports that are not connected to their appropriate port ?
No
Yes
Yes
Relabel the appropriate ports in that router
Select port not connected to its appropriate port call it corresponding_port
Is current_port already connected?
Yes
Disconnect current_port from existing connection
No
Is corresponding_port already connected?
Yes
Disconnect corresponding_port from existing connection
Connect current_port to corresponding_port
No

Fig. 11

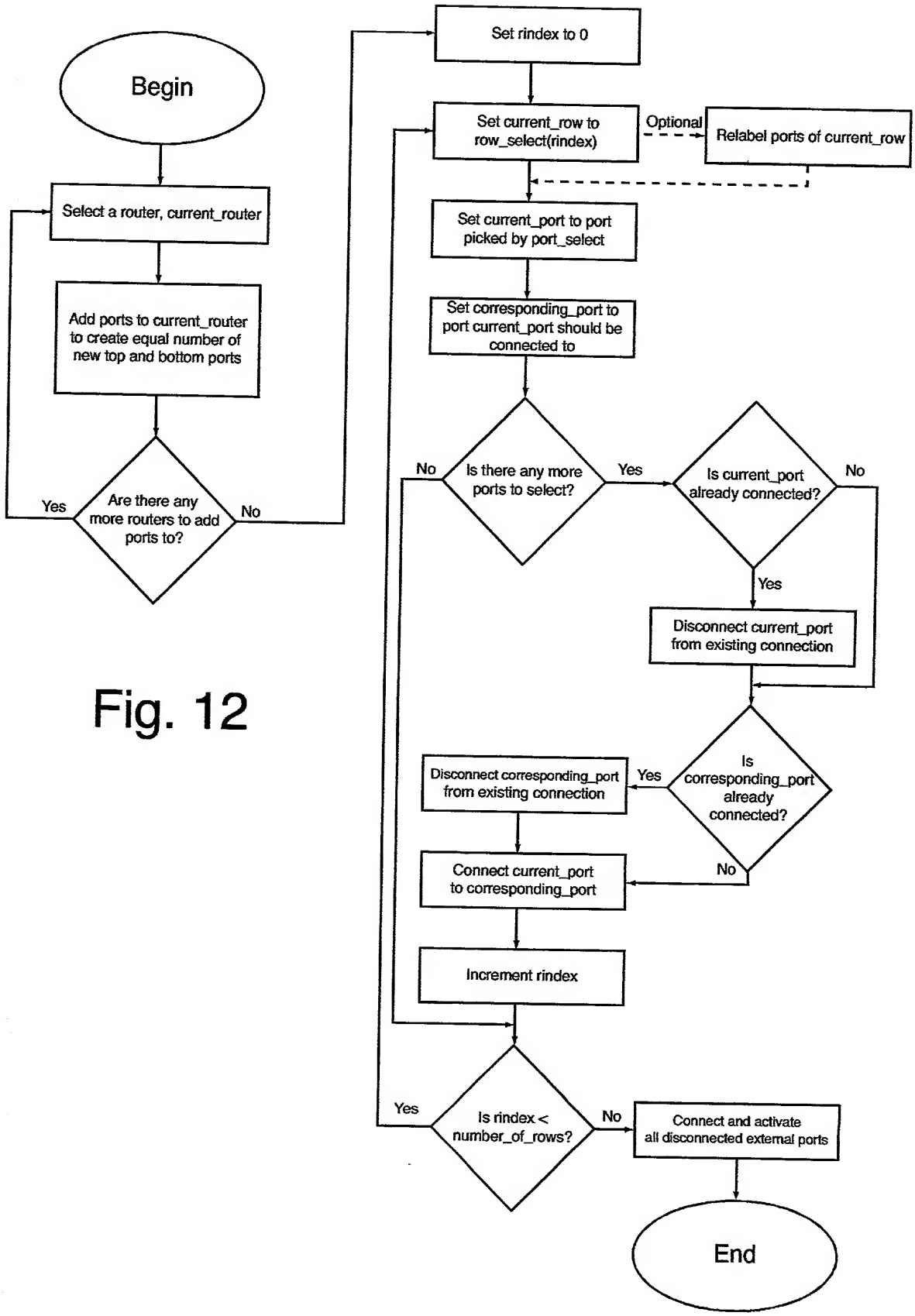



Fig. 12

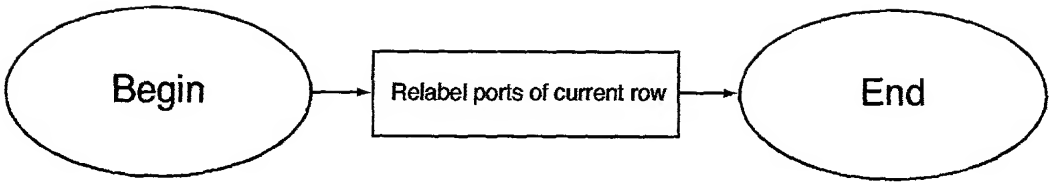



Fig. 13A

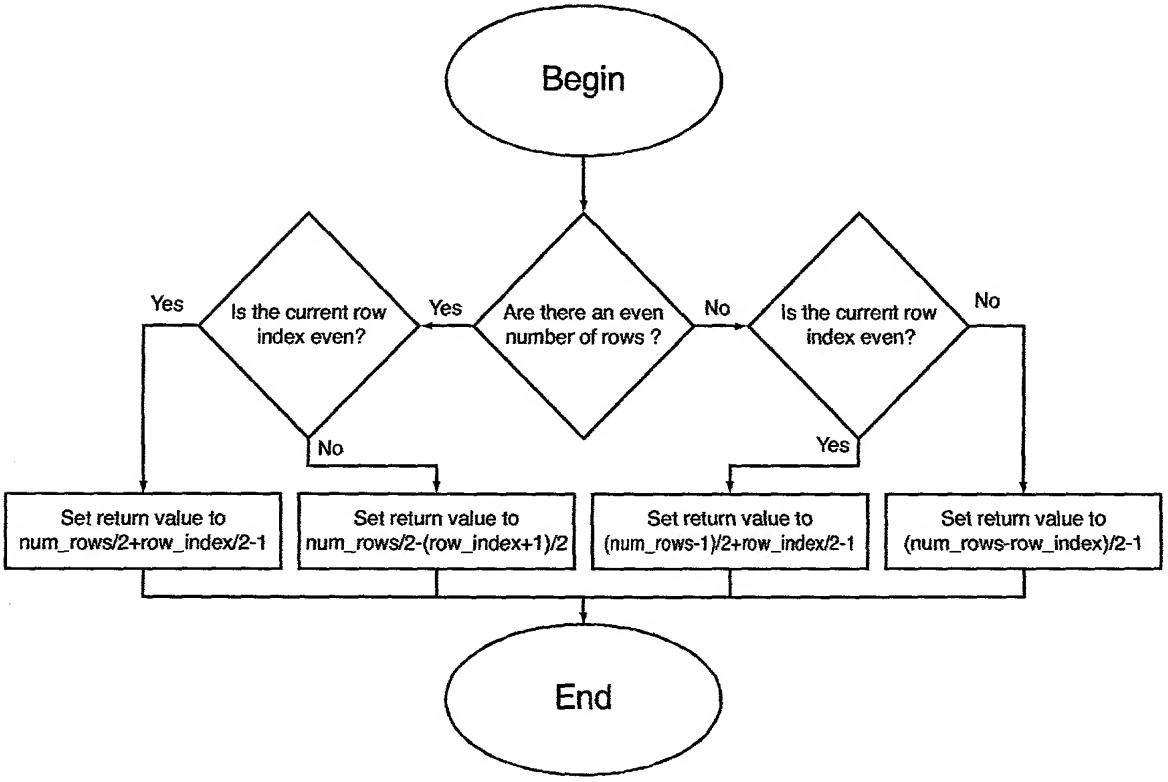



Fig. 13B

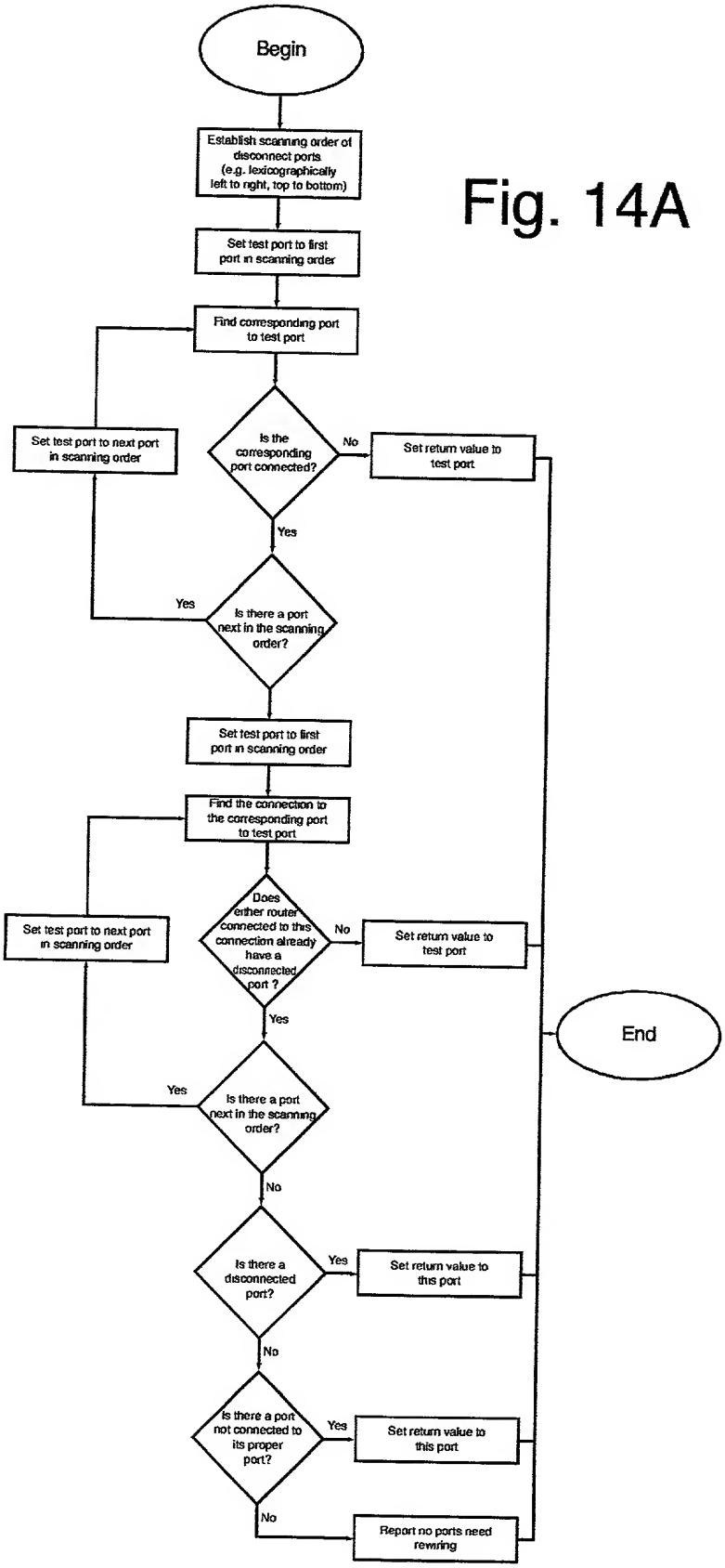



Fig. 14A

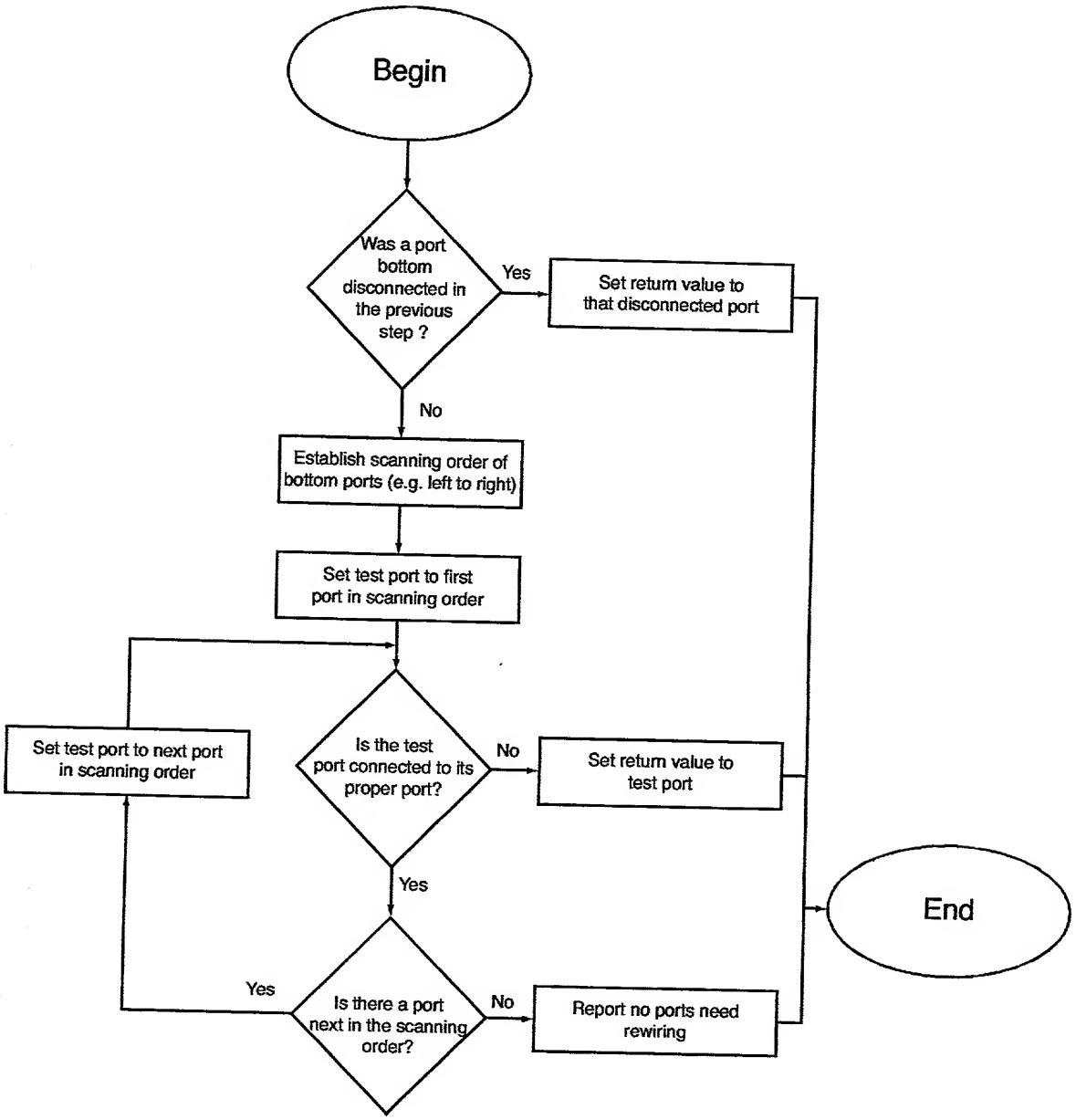
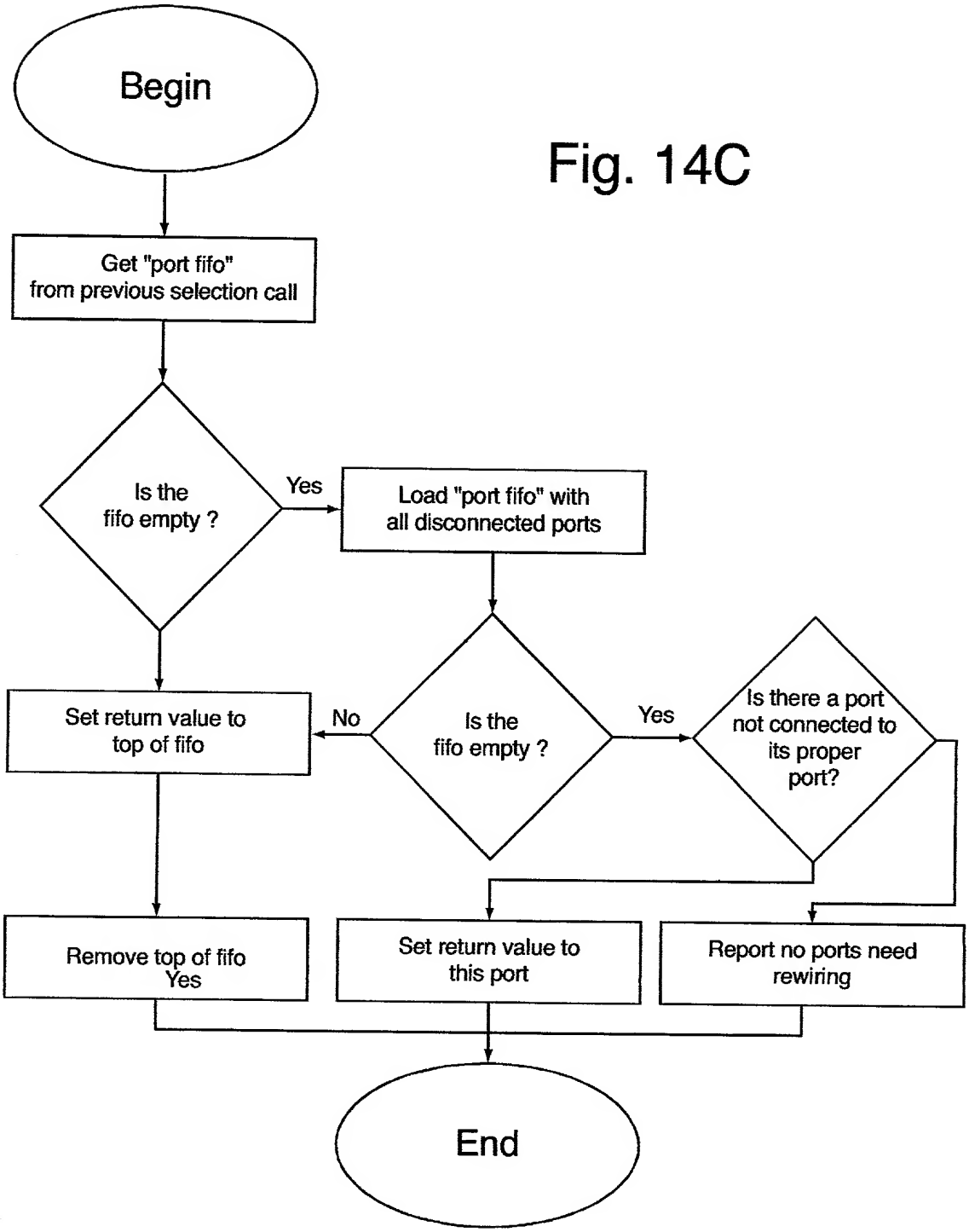

Begin
Was a port bottom disconnected in the previous step ?
Yes
Set return value to that disconnected port
No
Establish scanning order of bottom ports (e.g. left to right)
Set test port to first port in scanning order
Is the test port connected to its proper port?
No
Set return value to test port
Yes
Set test port to next port in scanning order
Is there a port next in the scanning order?
Yes
No
Report no ports need rewiring
End

Fig. 14B

Fig. 14C



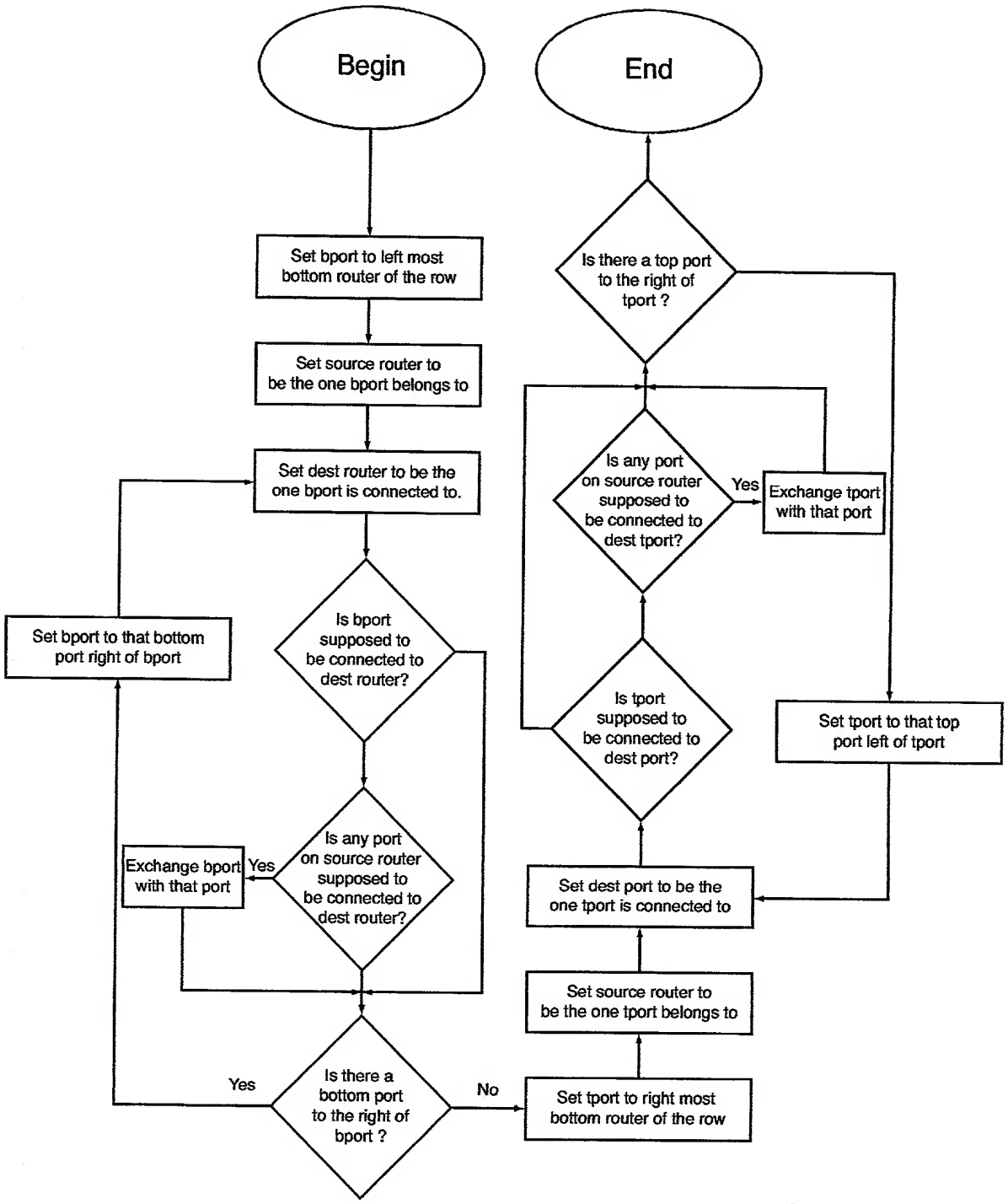



Fig. 15